International Journal of Mechanical and Production Engineering Research and Development (IJMPERD)
ISSN(P): 2249-6890; ISSN(E): 2249-8001
Vol. 7, Issue 1, Feb 2017, 35-40

SOFTWARE STATIC ANALYSIS FOR THE DEVELOPMENT OF AUTOMOTIVE ELECTRONIC ENGINE-MOUNT SOFTWARE

JEONGHYUN CHO[1], CHEOLWOONG AHN[2] & HYUNKI RYU[3]

[1]*Department. of Computer Engineering, Yeungnam University College, DaeGu, Korea*

[2]*Department of Smart Web Contents, Keimyung College University, DaeGu, Korea*

[3]*The GyeongBuk Research Institute of Vehicle Embedded Technology, Kyungpook, Korea*

ABSTRACT

In developing Automotive Software, the developer can be preventing in the logical defect of software by using a standard coding rules. The Coding guideline of MISRA-C is a coding rule that assures a high-reliability and high-quality. MISRA-C coding rule is required by automotive, airplane, manufacture and industry. In this paper, we will be applying the MISRA-C to target the software code of Active Control Engine Mount. And we will show a problems and solution that occurred after applying coding rule. As a result, it is expected that a software developer makes a high-reliability/high-quality of automotive software by reducing potential error of automotive software.

KEYWORDS : MISRA-C, Automotive, Static Analyse, Software & Coding rule

Received: Jan 01, 2017; **Accepted:** Jan 21, 2017; **Published:** Jan 31, 2017; **Paper Id.:** IJMPERDFEB20174

Original Article

INTRODUCTION

Recently, almost vehicles are equipped with an array of electronics designed to increase their safety, convenience, and eco-friendliness. An electric control unit (ECU) is responsible for user convenience and safety in the use of such automotive electronics. Owners of the latest-model vehicles now have access to a variety of services [1][2] that require the synchronization of ECUs in a vehicle, so software installed in ECUs is on the rise. In many case, OSEK OS [3][4][12] which is the automotive operating system, is applied also. New functions made available in vehicles by advanced electronics are increasing software testing cost. Recall of vehicles for repair of electronics issues is taken seriously.

After a large number of defects and malfunctions of vehicles have been found to be software issues, much effort is being made to prevent such issues by achieving a systematic improvement in software quality at the initial phases of development. A solution to such issues that has recently been discovered is static analysis of software at the commencement of development to minimize the chances of error [5][6]. This method does not require a high level of maturity of the software developer, and effectively prevents issues and achieves reliability at a relatively low cost.

In this study, an outline of cases where the MISRA-C 2004 coding guidelines [7][8] were applied to active electronic control engine mount (AECM) source codes using the quintessential static analysis tool QAC [9] will be given.

In Chapter 2 of this paper related research is introduced; static analysis methods and tools, and MISRA-C 2004 are outlined; and arithmetic type conversion, one of the coding guidelines of MISRA-C 2004, is explained.

In Chapter 3, cases of application of MISRA-C 2004 coding guidelines to program source codes are introduced. In Chapter 4, a conclusion is given and future research projects are outlined.

RELATED WORKS

- **Electronic Control Engine Mount**

The role of an engine mount is support of the powertrain in the car body and insulation of excitation force transferred to the car body. Stable support of the powertrain requires high rigidity of the engine mount, whereas effective insulation requires low rigidity of the engine mount; this design discrepancy must be accommodated by the engine mounting system.

The different types of engine mounts include the conventional mount made of rubber, the hydro mount with latex for powertrain movement control, the electronic active control mount designed to adjust mount rubber rigidity to suit driving conditions, and the ER/MR mount that uses electro-rheological fluid and magneto-rheological fluid in addition to the AECM technology for more precise control. An electronic AECM offsets vibration generated by a hybrid car engine and transferred to car body by generating out-of-phase driving force for improved ride comfort. Such an electronic AECM is made up of an actuator that generates driving force from inside the mount and a controller that supplies power and receives vibration signals for starting the actuator. An AECM and its controller, which includes control algorithms, are collectively referred to as an electronic control engine mount system.

- **Methods and Tools of Static Analysis of Code**

Defects in software that controls a vehicle are not just a matter of malfunction, but go so far as to cause injury or death, and even social issue. Needless to say, tremendous time and effort are invested into the prevention of such software defects. Despite resolution of serious software defects that took many years, automotive software is still being found with issues [10][11][13].

Given the gravity of software issues, the effort made by software developers to identify potential error codes or errors at the initial phases of software development is an important one.

There are two methods of analysis for finding defects or errors in program codes: static analysis and dynamic analysis. Static analysis breaks down the meaning of codes without executing codes to detect errors. Dynamic analysis compares predicted result of code execution with actual result of code execution to detect errors. Static analysis can be performed either by reading of codes by a developer or verifier, or by the use of an automated tool. Use of an automated static analysis tool enables source code analysis without program execution, and verification of adherence of analysed source codes to coding standards. Automated static analysis tools are frequently used for a more rigorous application of standards development methods and coding rules in a project. Common automated static analysis tools include LDRA, Code-Sonar, Prevent, QAC, and Poly-space.

- **MISRA-C 2004**

In the automotive, aerospace, railroad, control technology, and national defence industries where safety is prioritized, great effort is made to reduce software defects and errors. In the automotive industry, the Motor Industry Software Reliability Association promulgated the ISO C-based MISRA-C as the coding guidelines for the "C" programming language.

MISRA-C was announced in 1998 under the title "Guidelines for the Use of the C Language in Vehicle-based Software." In 2004, it was re-released as the "Guidelines for the Use of the C Language in Critical Systems" for application to other safety-prioritized systems in addition to vehicles.5) The purpose of MISRA-C is reduction of software defects and errors, as well as the achievement of safety, compatibility, and reliability of the codes of embedded systems created using the ISO C language.

The two application priority levels of MISRA-C are "required" and "advisory." MISRA-C 1998 includes 127 coding rules, of which 93 are required and 34 are advisory. MISRA-C 2004 includes 142 coding rules, of which 122 are required and 20 are advisory.

Figure 1: MISRA-C 2004 Coding Guidelines

CASE STUDY

- **Target and Environment of Application**

In this study, an automated static analysis tool was used to test the adherence of AECM source codes developed by Gyeongbuk Research Institute of Vehicle Embedded Technology to the coding standards. The testing and development environment used was "Freescale Code Warrior IDE."

The MCU was used in the controller of the driving motor of an electric vehicle. This test verifies adherence to Rule 10 of MISRA-C 2004, "Arithmetic Type Conversion."

- **Result of Tool Test**

Figure 2 is shown the result of QAC tool analyse. And, result of MISRA-C 2004 application, which is of the highest importance, is shown in Table 1.

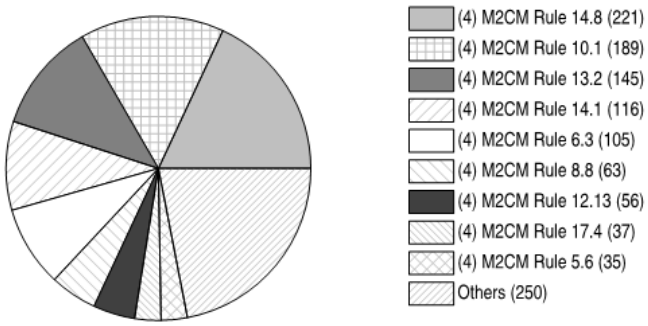

Figure 2: Result of MISRA-C 2004 Coding Rule Analyse

Table 1: Result of Violated Rules

Code Review Reference	MISRA-C: 2004 Reference	Code Review Message
M10.1	Rule 10.1	Implicit conversion of a complex integer expression to a smaller sized integer is not allowed.
M10.1.1	Rule 10.1	Implicit conversion of an integer expression to a different signedness is not allowed.
M14.8.3	Rule 14.8	The “do..while” statement should contain a compound statement {}.
M14.8.4	Rule 14.8	The “for” statement should be followed by a compound statement {}.

The following is a description of the key coding rules.[5)]

MISRA-C 2004 Rule 10.1: Conversion between different arithmetic types is liberal with C language, but type conversion of indistinct integers is not allowed.

In general, coding rule 10.1 is accompanied by 10.2.

MISRA-C 2004 Rule 10.2: If the below conditions present, a floating-point value must not be converted to another type discreetly.

Not a conversion to a floating-point value larger than (a)

- is a compound type
- is the actual argument of a function
- is a return type

Coding rule 10.2 applies to the type conversion of the floating point. It dictates that discreet type conversion that may result in data loss must not be performed. The following is a portion of a kernel source code in violation of coding rule 10.2.

```
#define CPU_CLOCK_FREQUENCY 40.0
#define PIT_BASE_CNT_VAL
(CPU_CLOCK_FREQUENCY)
PITMTLD0 = PIT_BASE_CNT_VAL;.
```

The above sample code violates the coding rule by assigning a "double" value to an "unsigned char" variable. To avoid this violation, the following explicit type casting is required.

```
#define CPU_CLOCK_FREQUENCY 40.0
#define PIT_BASE_CNT_VAL
(CPU_CLOCK_FREQUENCY)
PITMTLD0=
(double)PIT_BASE_CNT_VAL;
```

MISRA-C 2004 Rule 14.8: In the use of the C language, the terms "for," "do," and "while" must include component braces.

```
for (i=0 ; i<N_ELEMENTS; ++i)
{
buffer[i] = 0 ;
}
```

```
while (new_data_available)
process_data ();
service_whatchdog ();
```

The above sample recommends that the {} component be set in the term "while."

CONCLUSIONS

This study expounded the use of an automated tool for an arithmetic type conversion of vehicle AECM source codes in adherence to MISRA-C 2004. Moreover, issues arising from the application of coding rules related to the arithmetic type conversion of MISRA-C 2004 to actual vehicle AECM source codes and their solutions were expounded.

In future research errors and issues associated with software development will be analysed by using MISRA-C 2004 coding rules not mentioned in this paper, as well as the other software analysis method of dynamic analysis. The results of actual improvement will be duly implemented.

ACKNOWLEDGMENTS

This research was financially supported by the Ministry of Trade, Industry and Energy (MOTIE) and Korea Institute for Advancement of Technology(KIAT) through the Infrastructure Project for Industrial Technology Development (No.N0001156).

REFERENCES

1. *Seongsoo Hong, Jiyoung Park, Wooseok You, "Technology Trends in Automotive OS and Middleware: OSEK and AUTOSAR", Journal of the Korean Society for Precision Engineering Vo1.23, No 9, pp. 31-38, 2006.*
2. *Axel Dold, Daimler AG, "Implementation of requirements from ISO 26262 in the development of E/E components and systems", http://www.eacexpo.com/forum_2008/pdf/day_1/axeldold.pdf, Automotive Electronics and Electrical Systems Forum 2008 May 6, 2008, Stuttgart, Germany.*
3. *OSEK/VDX, "General Information", http://www.osek-vdx.org*
4. *OSEK/VDX, "Operation System Version 2.2.3", http://portal.osek-vdx.org*
5. *Robert N. Charette, "This Car Runs on Code", http://www.spectrum.ieee.org/greentech/advanced-cars/this-car-runs-on-code,February 2009.*
6. *Programming languages-C, http://www.open-std.org/JTC1/SC22/WG14/www/docs/n1539.pdf*
7. *Misra, "Misra-C 2004" http://misra.org.uk*
8. *MISRA (2004), MISRA C : 2004 : Guidelines for the Use of the C Language in Critical Systems, MISRA.*
9. *QAC, "PRQA QAC", http://www.programmingresearch.com/products/qac/*
10. *Jin-Hee Cho, Kyungmin Park, Tae-Man Han, Yangjae Jung, Seohyun Jeon and Hyeon Soo Kim, "An Analysis of ISO 26262 and its applications ", KSAE, 2009 Annual Conference, pp. 1691-1702.*
11. *Robert N. Charette, "This Car Runs on Code", http://www.spectrum.ieee.org/greentech/advanced-cars/this-car-runs-on-code,February 2009*
12. *Sungrae Cho, Hyunki Ryu, Sungho Jin and Junho Lee, "An Implementation of Automotive Operation System based on OSEK/VDX and Methods for Ensuring the Reliability of that Implementation", KSAE 2009 Annual Conference, pp. 1816-1822.*

13. Jaejin Baek, "A Study on Reliability Evaluation of Embedded Software in Vehicle", KSAE, Vol. 19, No. 4, pp.1-7, 2011.

AUTHOR PROFILE

Jeong-Hyun, Cho

Jeong-Hyun, Cho received the BS degree in Electrical Engineering and Computer Science from the Kyungpook National University, MS and PhD degree in computer engineering from the Kyungpook National University in Korea. He is with the Dept. of Computer Engineering, Yeungnam University College.

CheolWoong, Ahn

CheolWoong, Ahn received the BS degree in Electrical Engineering and Computer Science from the Kyungpook National University, MS and PhD degree in computer engineering from the Kyungpook National University in Korea. He is with the Dept. of Smart Web Contents, Keimyung College University.

Hyun-Ki, Ryu

Hyun-Ki, Ryu received the BS degree in Computer engineering from the Dong-AUniversity, MS degree in computer engineering from the Kyungpook National University,and the PhD Candidate in electrical engineering from the Kyungpook National University in Korea, He is a researcher at the Kyungpook Research Institute of Vehicle Embedded Technology in Korea. His research interests include vehicle embedded system and image processing algorithm.